Manual de usuario

Analizador Sintáctico

Lenguaje NESP

Compiladores

Integrantes:

Daniel Eduardo Ruiz C.

22 de Marzo, 2025

Periodo 2025a

Tabla de Contenidos

Introducción

Para el análisis léxico o conversión de código fuente a tokens, así como el análisis sintáctico (la generación del árbol sintáctico) de un lenguaje, de programación en español que pretende tener coherencia lógica y ser más cercano al lenguaje natural, así como estar al mismo tiempo en español para que sea más amigable con los usuarios nuevos que tengan que aprender programación y no sepan inglés, de esta forma se consigue mayor facilidad al hacer la transición al lenguaje estándar con raíces en el inglés. En el ánimo de crear un compilador para el mismo usando ANTLR en su versión para python.

Descripción

NESP, este lenguaje se basa en utilizar diversas expresiones estructuras en español con relativa facilidad y semejanza a oraciones simples que pretende emular la fase de desarrollo cuando se estructura un algoritmo, reemplazando palabras clave y operadores tradicionales por términos en español, con el objetivo de facilitar el razonamiento, notar que principalmente se hizo uso de los signos de puntuación para darle coherencia, usando el punto para separar instrucciones como si de oraciones se tratara y definiendo correctamente el signo de igual como asignación (no igualdad matemática), y el signo doble de igual cómo “es igual que”, etc, todo ello consigue un mejor entendimiento de lo que de verdad hace un lenguaje, así como facilitar la transición de un algoritmo escrito a lenguaje de programación.

Componentes principales:

- gramaticaEspanol.g4: Gramática del analizador léxico.
- parserGramaticaEspanol.g4: Gramática del analizador léxico.
- analizador_lexico.py: Script de python para usar el analizador (descartado).
- analizador_sintactico.py: Script de python para usar el analizador más reciente.

Instalación y uso

1. Requisitos

a. Python >= 3.6
b. ANTLR4 Runtime para python.
c. Graphviz para la generación imagen del árbol

2. Ejecutar el siguiente script:

```
pip install antlr4-tools
pip install antlr4-python3-runtime
antlr4 -Dlanguage=Python3 archivos/gramaticaEspanol.g4
antlr4 -Dlanguage=Python3 -visitor archivos/parserGramaticaEspanol.g4
```

*Comentar las siguientes líneas [7-8] en los archivos generados:
“parserGramaticaEspanol .py” y “gramaticaEspanol.py”:

3. Para analizar un archivo.

```
#  Código fuente .nesp
python analizador_sintactico.py ejemplo.nesp
# Texto directo
python analizador_sintactico.py -c “código fuente”
```

* Se generará una carpeta con el nombre del archivo y un timestamp, la cual contendrá:

1. Archivo .html con tres tablas (arból sintáctico, tabla de tokens, tabla de análisis sintáctico.
2. Archivo de grafo de árbol sintáctico.
3. Sólo generará una imagen si se usa texto directo debido al tamaño del árbol.

Características

Reglas generales:

1. Las funciones deben comenzar con funcion y finalizar con fin.
2. Los bloques condicionales (si) deben llevar entonces: y cerrarse con fin.
3. Los bucles (mientras) deben llevar entonces: y cerrarse con fin.

Palabras Reservadas:

- si ➜ Condicional (equivalente a if).
- entonces: ➜ Inicia bloque condicional y siempre debe ir seguido de :.
- fin ➜ Indica el fin de un bloque (} en otros lenguajes).
- mientras ➜ Bucle (while).
- asigna ➜ Operador de asignación (=).
- a la ➜ Operador de potencia (^).
- devuelve ➜ Retorno de función (return).
- funcion ➜ Definición de función (function).
- es un ➜ Declaración de tipo de dato.

Estructuras de Control:

- si condicion entonces: ➜ Inicia un bloque condicional.

- fin ➜ Indica el fin del bloque.
- mientras condición entonces: ➜ Inicia un bucle, finaliza con fin.

Definición de Funciones:

- Se declaran con funcion, seguidas del nombre y los parámetros entre paréntesis.
- El tipo de retorno se define con devuelve.
- El cuerpo de la función comienza después de : y termina con fin.

Tipos de Datos:

- entero ➜ Números enteros (int).
- real ➜ Números con decimales (float).
- texto ➜ Cadenas de caracteres (string).
- bit ➜ Booleano (true/false).
- arreglo de ➜ Estructuras multidimensionales (array).

Operadores Relacionales:

- es distinto de ➜ !=
- es igual que ➜ ==
- es mayor que ➜ >
- es menor que ➜ <
- es mayor o igual que ➜ >=
- es menor o igual que ➜ <=

Operadores Aritméticos:

- mas ➜ +
- menos ➜ -
- por ➜ *
- entre ➜ /

Operadores Lógicos:

- y ➜ &&
- o ➜ ||
- no ➜ !

Identificadores:

- Inician con una letra y pueden contener letras, números y guiones bajos (_).

Constantes y Literales:

- Números enteros y reales: 123, 3.14, -25.6
- Cadenas: Texto delimitado por comillas ("Hola mundo").

Delimitadores y Símbolos Especiales:

- (y) ➜ Paréntesis
- : ➜ Obligatorio después de entonces: y después de definir una función.
- . ➜ Fin de línea (equivalente a ;).
- fin ➜ Indica el fin de una estructura (} en otros lenguajes).
- [y] ➜ Corchetes indican que se recupera información de un arreglo.

Manejo de Espacios, Indentaciones y Comentarios:

- Los comentarios deben comenzar con Nota: y terminar con /.

- Se ignorar espacios e indentaciones repetidas.

Gramática NESP

A continuación se presenta la documentación detallada de la gramática utilizada para el lenguaje NESP. Esta documentación explica cada expresión y regla definida en los archivos de gramática ANTLR4.

1. Analizador Léxico (gramaticaEspanol.g4)

El analizador léxico define los tokens (unidades léxicas) que componen el lenguaje.

Palabras Reservadas

Token	**Expresión**	**Descripción**
FUNCION	'funcion'	Define el inicio de una función
SI	'si'	Inicia estructura condicional
ENTONCES	'entonces:'	Marca el inicio del bloque condicional
FIN	'fin'	Termina bloques estructurados (funciones, condicionales)
MIENTRAS	'mientras'	Inicia estructura iterativa
ASIGNA	'asigna'	Operador de asignación
A_LA	'a la'	Operador para exponentes

Token	Expresión	Descripción
DEVUELVE	'devuelve'	Retorno de función
ES_UN	'es un'	Declaración de variables
ARREGLO_DE	'arreglo de'	Declaración de arreglos
DE	'de'	Utilizado en declaración de arreglos

Operadores Relacionales

Token	Expresión	Descripción
ES_DISTINTO_DE	'es distinto de'	Operador de desigualdad (≠)
ES_IGUAL_QUE	'es igual que'	Operador de igualdad (=)
ES_MAYOR_QUE	'es mayor que'	Operador mayor que (>)
ES_MENOR_QUE	'es menor que'	Operador menor que (<)
ES_MAYOR_O_IGUAL_QUE	'es mayor o igual que'	Operador mayor o igual (≥)
ES_MENOR_O_IGUAL_QUE	'es menor o igual que'	Operador menor o igual (≤)

Operadores Aritméticos

Token	**Expresión**	**Descripción**
MAS	'mas'	Operador de suma (+)
MENOS	'menos'	Operador de resta (-)
POR	'por'	Operador de multiplicación (*)
ENTRE	'entre'	Operador de división (/)

Operadores Lógicos

Token	**Expresión**	**Descripción**
Y	'y'	Operador lógico AND
O	'o'	Operador lógico OR
NO	'no'	Operador lógico NOT

Tipos y Literales

Token	**Expresión**	**Descripción**
TIPO_DATO	'entero' | 'real' | 'texto' | 'bit'	Tipos de datos básicos
ID	`[a-zA-ZáéíóúÁÉÍÓÚñÑ][a-zA-Z0-9_áéíóúÁÉÍÓÚñÑ]*`	Identificadores (nombres de variables y funciones)

Token	Expresión	Descripción
NUMERO	`[0-9]+ ('.' [0-9]+)?`	Literales numéricos (enteros o decimales)
CADENA	`'"' (~["\r\n])* '"'`	Literal de texto (entre comillas dobles)

Comentarios y Delimitadores

Token	Expresión	Descripción
COMENTARIO	'Nota:' .*? '/' -> skip	Comentarios (ignorados en el análisis)
PARENTESIS_IZQ	'('	Paréntesis de apertura
PARENTESIS_DER	')'	Paréntesis de cierre
CORCHETE_IZQ	'['	Corchete de apertura (para arreglos)
CORCHETE_DER	']'	Corchete de cierre (para arreglos)
DOS_PUNTOS	':'	Delimitador de dos puntos
PUNTO	'.'	Delimitador de punto (fin de instrucción)
COMA	','	Delimitador de coma (separador)

Otros

Token	Expresión	Descripción
WS	`[ \t\r\n]+ -> skip`	Espacios en blanco (ignorados)
ERROR	`.`	Captura cualquier carácter no reconocido como error

2. Analizador Sintáctico (parserGramaticaEspanol.g4)

El analizador sintáctico define las reglas de estructura que forman el lenguaje.

Reglas Principales

Regla	Expresión	Descripción
programa	`instruccion+ EOF`	Programa completo (secuencia de instrucciones)
instruccion	`declaracion PUNTO` | `asignacion PUNTO` | `sentencia_if` | `sentencia_while` | `definicion_funcion` | `retorno PUNTO` | `llamada_funcion PUNTO` | `expresion PUNTO`	Elementos principales del programa

Declaraciones

Regla	Expresión	Descripción
declaracion	`ID ES_UN TIPO_DATO` | `ID ES_UN ARREGLO_DE TIPO_DATO DE dimension`	Define variables simples o arreglos
dimension	`NUMERO` | `NUMERO (POR NUMERO)+`	Dimensiones de arreglos (1D, 2D, etc.)

Asignaciones

Regla	Expresión	Descripción
asignacion	`designador ASIGNA expresion` | `ID ASIGNA condicion`	Asigna valores a variables o elementos de arreglo
designador	`ID` | `ID CORCHETE_IZQ lista_indices CORCHETE_DER`	Referencia a variable o elemento de arreglo

Estructuras de Control

Regla	Expresión	Descripción
sentencia_if	`SI condicion ENTONCES bloque FIN`	Estructura condicional (if)
sentencia_while	`MIENTRAS condicion ENTONCES bloque FIN`	Bucle while

Regla	Expresión	Descripción
bloque	`instruccion+`	Bloque de código (secuencia de instrucciones)

Funciones

Regla	Expresión	Descripción
definicion_funcion	`FUNCION ID PARENTESIS_IZQ parametros? PARENTESIS_DER DOS_PUNTOS bloque FIN`	Define una función
parametros	`ID (COMA ID)*`	Lista de parámetros de función
retorno	`DEVUELVE expresion`	Sentencia de retorno
llamada_funcion	`ID PARENTESIS_IZQ argumentos? PARENTESIS_DER`	Invocación de función
argumentos	`expresion (COMA expresion)*`	Lista de argumentos para llamada a función

Condiciones

Regla	Expresión	Descripción
condicion	`expresion operador_relacional expresion` | `expresion` | `condicion Y condicion` | `condicion O condicion` | `NO condicion` | `PARENTESIS_IZQ condicion PARENTESIS_DER`	Expresiones booleanas
operador_relacional	`ES_IGUAL_QUE` | `ES_DISTINTO_DE` | `ES_MAYOR_QUE` | `ES_MENOR_QUE` | `ES_MAYOR_O_IGUAL_QUE` | `ES_MENOR_O_IGUAL_QUE`	Operadores de comparación

Expresiones

Regla	Expresión	Descripción
expresion	`termino` | `expresion (MAS | MENOS) expresion` | `expresion (POR | ENTRE) expresion` | `expresion A_LA expresion` | `PARENTESIS_IZQ`	Cálculos y valores

Regla	Expresión	Descripción
	`expresion PARENTESIS_DER` | `llamada_funcion` | `ID CORCHETE_IZQ lista_indices CORCHETE_DER`	
termino	`ID` | `NUMERO` | `CADENA`	Elementos básicos de expresión
lista_indices	`expresion (COMA expresion)*`	Índices para acceso a arreglos

Ejemplo Archivo

```
Nota: Este es un programa de ejemplo para probar el analizador léxico y
sintáctico./

funcion factorial(n):
   si n es menor o igual que 1 entonces:
       devuelve 1.
   fin
   devuelve n por factorial(n menos 1).
fin

Nota: Ejemplo de declaración de variables y arreglos./
contador es un entero.
temperatura es un real.
nombre es un texto.
activo es un bit.

Nota: Declaración de arreglos según el formato especificado./
vector es un arreglo de entero de 10.
matriz es un arreglo de real de 3 por 3.
cubo es un arreglo de entero de 2 por 2 por 2.

Nota: Declaración de variables para almacenar valores de arreglos./
```

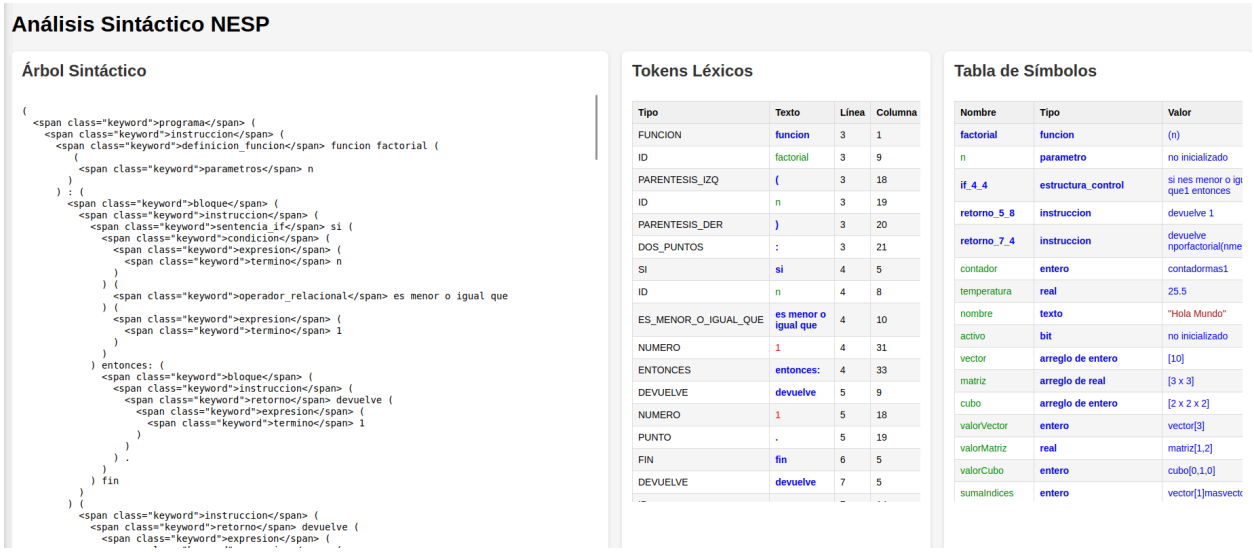
```
valorVector es un entero.
valorMatriz es un real.
valorCubo es un entero.
sumaIndices es un entero.


Nota: Ejemplo de asignación y operaciones./
contador asigna 1.
mientras contador es menor que 10 entonces:
   vector[contador] asigna contador por 5.
   Nota: Asignación a un elemento del vector./
   contador asigna contador mas 1.
   si contador es igual que 5 entonces:
       Nota: Esto es un comentario dentro del bloque./
       temperatura asigna 25.5.
   fin
fin

resultado asigna 2 a la 3.
nombre asigna "Hola Mundo".

Nota: Nuevos ejemplos de recuperación de valores de arreglos./
valorVector asigna vector[3].             Nota: Recupera el valor en el índice
3 del vector./
matriz[1, 2] asigna 9.81.                  Nota: Asigna un valor a un elemento
de la matriz./
valorMatriz asigna matriz[1, 2].           Nota: Recupera el valor en la fila
1, columna 2 de la matriz./
cubo[0, 1, 0] asigna 42.                   Nota: Asigna un valor a un elemento
del cubo./
valorCubo asigna cubo[0, 1, 0] .           Nota: Recupera el valor en las
coordenadas 0, 1, 0 del cubo./

Nota: Usando acceso a arreglo dentro de una expresión./
sumaIndices asigna vector[1] mas vector[2]. Nota: Suma los valores de dos
elementos del vector./
```



Análisis Sintáctico NESP

Árbol Sintáctico

```
(
  <span class="keyword">programa</span> (
    <span class="keyword">instruccion</span> (
      <span class="keyword">definicion_funcion</span> funcion factorial (
          (
            <span class="keyword">parametros</span> n
          )
        ) : (
          <span class="keyword">bloque</span> (
            <span class="keyword">instruccion</span> (
              <span class="keyword">sentencia_if</span> si (
                <span class="keyword">condicion</span> (
                  <span class="keyword">expresion</span> (
                    <span class="keyword">termino</span> n
                  )
                ) (
                  <span class="keyword">operador_relacional</span> es menor o igual que
                ) (
                  <span class="keyword">expresion</span> (
                    <span class="keyword">termino</span> 1
                  )
                )
              ) entonces: (
                <span class="keyword">bloque</span> (
                  <span class="keyword">instruccion</span> (
                    <span class="keyword">retorno</span> devuelve (
                      <span class="keyword">expresion</span> (
                        <span class="keyword">termino</span> 1
                      )
                    )
                  ) .
                )
              ) fin
            )
          ) (
            <span class="keyword">instruccion</span> (
              <span class="keyword">retorno</span> devuelve (
                <span class="keyword">expresion</span> (
```

Tokens Léxicos

Tipo	Texto	Línea	Columna
FUNCION	funcion	3	1
ID	factorial	3	9
PARENTESIS_IZQ	(	3	18
ID	n	3	19
PARENTESIS_DER	)	3	20
DOS_PUNTOS	:	3	21
SI	si	4	5
ID	n	4	8
ES_MENOR_O_IGUAL_QUE	es menor o igual que	4	10
NUMERO	1	4	31
ENTONCES	entonces:	4	33
DEVUELVE	devuelve	5	9
NUMERO	1	5	18
PUNTO	.	5	19
FIN	fin	6	5
DEVUELVE	devuelve	7	5

Tabla de Símbolos

Nombre	Tipo	Valor
factorial	funcion	(n)
n	parametro	no inicializado
if_4_4	estructura_control	si nes menor o igu que1 entonces
retorno_5_8	instruccion	devuelve 1
retorno_7_4	instruccion	devuelve nporfactorial(nme
contador	entero	contadormas1
temperatura	real	25.5
nombre	texto	"Hola Mundo"
activo	bit	no inicializado
vector	arreglo de entero	[10]
matriz	arreglo de real	[3 x 3]
cubo	arreglo de entero	[2 x 2 x 2]
valorVector	entero	vector[3]
valorMatriz	real	matriz[1,2]
valorCubo	entero	cubo[0,1,0]
sumaIndices	entero	vector[1]masvectc

Página html generada para el archivo de texto.

Ejemplo Texto

```
python src_main/analizador_sintactico.py -c "matriz es un arreglo de real de 3 por 3. matriz[0, 0] asigna 9.81."
```

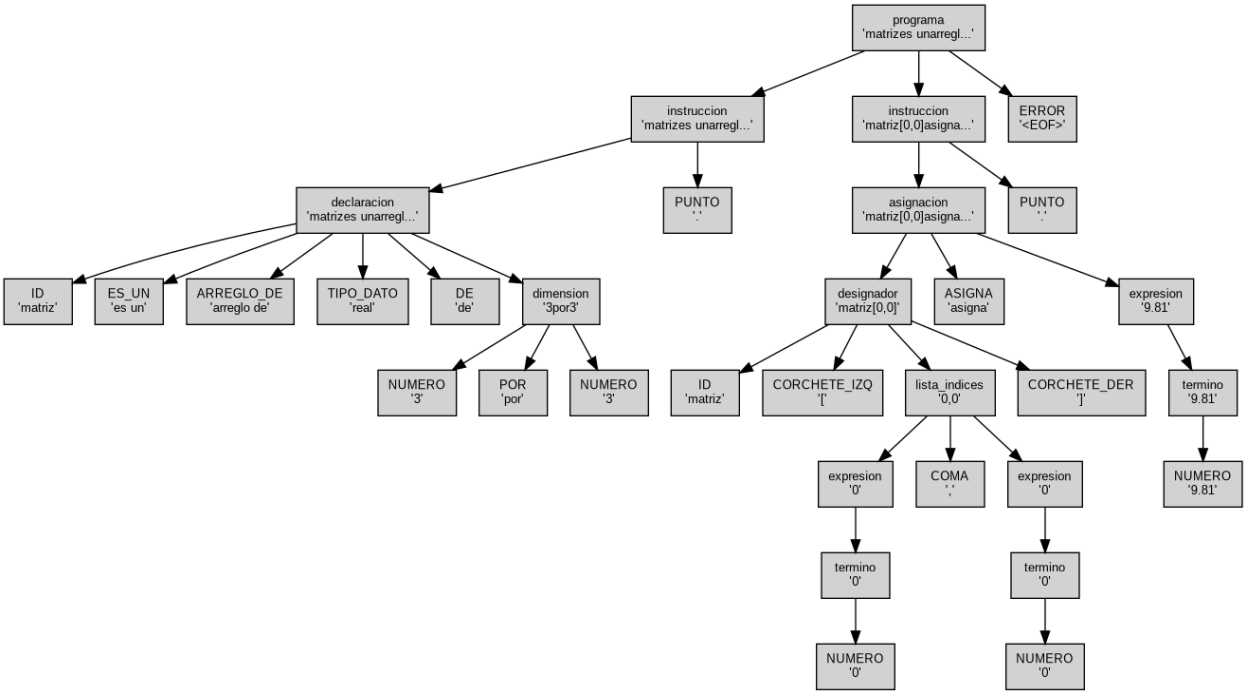
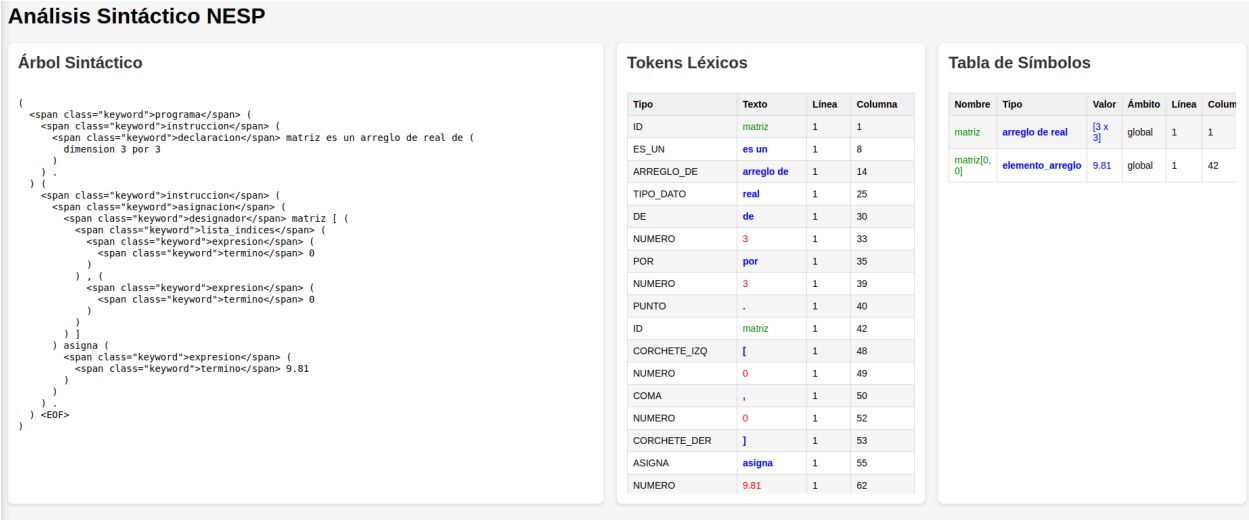


Imagen generada del árbol sintáctico.



Análisis Sintáctico NESP

Árbol Sintáctico

```
(
  <span class="keyword">programa</span> (
    <span class="keyword">instruccion</span> (
      <span class="keyword">declaracion</span> matriz es un arreglo de real de (
        dimension 3 por 3
      )
    ) .
  ) (
    <span class="keyword">instruccion</span> (
      <span class="keyword">asignacion</span> (
        <span class="keyword">designador</span> matriz [ (
          <span class="keyword">lista_indices</span> (
            <span class="keyword">expresion</span> (
              <span class="keyword">termino</span> 0
            )
          ) , (
            <span class="keyword">expresion</span> (
              <span class="keyword">termino</span> 0
            )
          )
        ) ]
      ) asigna (
        <span class="keyword">expresion</span> (
          <span class="keyword">termino</span> 9.81
        )
      )
    ) .
  ) <EOF>
)
```

Tokens Léxicos

Tipo	Texto	Línea	Columna
ID	matriz	1	1
ES_UN	es un	1	8
ARREGLO_DE	arreglo de	1	14
TIPO_DATO	real	1	25
DE	de	1	30
NUMERO	3	1	33
POR	por	1	35
NUMERO	3	1	39
PUNTO	.	1	40
ID	matriz	1	42
CORCHETE_IZQ	[	1	48
NUMERO	0	1	49
COMA	,	1	50
NUMERO	0	1	52
CORCHETE_DER	]	1	53
ASIGNA	asigna	1	55
NUMERO	9.81	1	62

Tabla de Símbolos

Nombre	Tipo	Valor	Ámbito	Línea	Colum
matriz	arreglo de real	[3 x 3]	global	1	1
matriz[0, 0]	elemento_arreglo	9.81	global	1	42

Página html generada con 3 tablas.

Conclusión

Se desarrollará el análisis semántico final partiendo del árbol sintáctico generado en esta fase, además de servir como una referencia para posteriores actualizaciones además sería interesante agregar algún paradigma como programación orientada a objetos o a prototipos, funcional , etc. Pero por el momento sólo hay funciones, no se pueden definir objetos ni clases.